REMARKS

This paper is responsive to the Office Action mailed December 21, 2006. Filed herewith is a Request for a Three Month Extension of Time, which extends the statutory period for response to expire on June 21, 2006. Accordingly, Applicant respectfully submits that this response is being timely filed.

Claims 1-12 are pending in this application. Claims 1, 2 and 4 have been amended. Claims 3 and 11 have been cancelled. New claims 14 and 15 have been added.

Claims Rejections – 35 USC § 101

At ¶1, the Examiner rejects claims 1-4 under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter. Although the Applicant disagrees with the Examiner's assessment, claims 1, 2 and 4 have been amended to overcome this rejection. Claim 3 has been canceled.

In particular, claims 1-4 have been amended to "breath life" into the intended use recited in the preamble of each claim, as requested by the Examiner. However, the Applicant does not agree with the Examiner's statement that, "transforming numbers (memory access addresses) to produce 'transformed' memory access addresses does not, in and of itself, achieve a useful, concrete, and tangible result." For example, the M.P.E.P at section 2106.II.A cites *State Street Bank & Trust Co. v. Signature Financial Group Inc.* as follows:

> "[T]ransformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces 'a useful, concrete and tangible result' -- a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades." *State Street*, 149 F.3d at 1373, 47 USPQ2d at 1601

Similarly, in the rejected claims, transforming addresses conforming to one convention to addresses conforming to a second convention is clearly a useful, concrete and tangible result – namely, a set of addresses that can be used by a processor that observes the second convention.

Thus, the Applicant believes that the original claims already go beyond merely "transforming memory access addresses" as an abstract activity. However, the amendments reinforce this by the introducing a second step of "issuing said instruction for execution by said processor of the second type" to address the Examiner's concerns.

Claims Rejections – 35 USC § 112

At ¶2, the Examiner rejects claims 2 and 10 under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement. We are pleased to note that the previous objections under 35 USC § 112 have been withdrawn.

We respectfully disagree with the Examiner's current rejection of claims 2 and 10. Regarding claim 2, the Examiner states that this claim recites steps that are not operative for the stated purpose of "use in emulating a processor." However, the method of claim 2 is for use in emulating a first processor using a second processor. Such emulation involves the transformation of program code from the first processor (subject system) to run on the second processor (target system). In generating the target instructions, the transformation process has complete control over the manner in which those instructions will be executed. The Examiner correctly notes that the hardware of the target system will automatically increment the target program counter. The claimed method concerns the effect of executing those target instructions – namely that the memory access addresses provided in the subject instructions are transformed as claimed. As the application makes clear, such transformed memory access addresses apply to a plurality of bytes within memory as appropriate to the subject program. Hence, the target program counter can indeed automatically increment as is usual for the target hardware, but still the memory addresses affected by executing those target instructions address the reverse relative order of words as claimed. The "inventions and modifications" to the second processor suggested by the Examiner are not necessary. If this point still causes difficulty with the Examiner then the Applicant would be very pleased to address it in a personal interview.

Claims Rejections – 35 USC § 102

At ¶3, the Examiner rejects claims 1, 3-9, and 11-12 have been rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 5,968,164 to Loen et al. (Loen). We are pleased to note that the objections raised previously against claims 2 and 10 have been withdrawn.

Regarding the rejection of claim 1, we feel that the original wording of claim 1 clearly set forth novel features over the disclosure of Loen. However, the amendments to claim 1 further reinforce those differences. Loen provides a specially adapted form of hardware which includes "reflection circuitry" 121 of Figure 1, which reverses the order of bytes within a word during a fetch from memory. Loen performs a modification of the address offset within a word using the hardware address offset modification circuitry 119 of Figure 1, so that the address offset still points to the correct byte within the word, even though the order of the bytes within the word has been reflected. That is, the hardware of Leon transforms byte order within a data word, and adjusts a memory address offset on a word by word basis as part of the fetch and store pathway. Loen does not receive instructions, transform memory access addresses with reference to those instructions, and then issue instructions for execution by the second processor, where those issued instructions include the transformed memory access addresses, as claimed.

In summary, Loen does not transform memory access addresses with reference to instructions, but instead applies a hardware transformation to data words within the fetch and store pathway. Further, Loen only manipulates the bytes within a data double word or part thereof. Loen does not rearrange the words, but the transformation of the claimed invention does.

Although claim 3 has been canceled, we respectfully disagree with the Examiner's analysis concerning claim 3 for the reasons set forth above with respect to claim 1.

Concerning the rejection of claim 4, we appreciate the Examiner's position and we are grateful for the additional example provided by the Examiner. We feel that the key point here is that Loen provides hardware that performs byte reflection and address offset modification, on a word-by-word basis, as part of the fetch and store pathway. That is, the byte reflection and address offset modification are applied to each word individually in turn. Loen does not provide a transformation that applies to a plurality of words. We submit that the language "relating to a

plurality of words" is sufficient to bring out this distinction between performing a task individually but repeated a number of times, compared with performing a task in relation to a whole group. For completeness, we also note that claim 4 reinforces the environment of the claimed invention wherein the memory access addresses of received instructions are transformed, and the transformed memory access addresses are provided in the output instructions for execution on the second processor. Loen does not transform addresses within instructions.

Concerning the rejection of claim 5, Loen does not disclose "including the thus changed address reference in a compiled or translated output instruction" as claimed. Loen is operating at the hardware level with the reflection circuitry 121 and the address offset modification circuitry 119. As stated at column 6, line 27-29 "when the circuitry is initialized to operate in the alternate endian mode, reflection of the data takes place." The circuitry of Loen is part of the fetch and store pathway, downstream from decoding a particular instruction. As noted by the Examiner at column 7, line 62 to column 8, line 18, "to complete the processor fetch from memory, an address [offset] modification is performed on the address [offset] as originally presented by the software." Loen reflects the data within a particular word, when that word is fetched into the processor. Loen does not include changed address references in a compiled or translated output instruction as in claim 5. Further, claim 5 recites that the changed address reference is included "such that there is no extra operation required during execution of the output instruction to accommodate the [alternate endian] convention". In claim 5, this feature provides a clear distinction over Loen where the reflection circuitry 121 and the address offset modification circuitry 119 are clearly required during execution of the instruction.

Regarding the rejection of claim 6, Loen does not disclose a method for use in compilation or translation of computer program source code, noting the dependency of claim 6 on claim 5.

Concerning the rejection of claim 7, as we have discussed above, it is common ground between the Applicant and the Examiner that in the prior art, words are addressed in the same order for both big endian and little endian hardware. By contrast claim 7 requires that the fixed block of memory be addressed from a predetermined one of its two ends depending on the endian convention. Loen does not disclose this step of determining which of the available two ends should be utilized.

Concerning the rejection of claim 8, as discussed above Loen does not disclose changing memory access addresses so that the fixed block of memory contents is inverted – noting "a plurality of words" in claim 5. In Loen, the order of words is not inverted.

Regarding the rejections of claims 9 and 12 are allowable for similar reasons to claims 1 and 4, as set forth above.

Although claim 13 has been canceled, we respectfully disagree with the Examiner's analysis concerning claim 13 for the reasons set forth above with respect to claim 1.

Claims Rejections – 35 USC § 103

Claims 2 and 10 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over "Data Structures & Other Objects Using C++," by Michael Main and Walter Savitch (Savitch). Firstly, taking the Examiner's analogy that one letter equals one byte, the stack of Savitch only reverses the relative order of bytes (letters) within a word. Savitch does not disclose reversing the order of words.

Secondly, there is no motivation to combine the teachings of Savich with the teaching of Loen, since combining the stack of Savitch with the hardware of Loen is illogical. Savitch concerns "data structures and other objects using C++" as a high level programming language. Claims 2 and 10 concern transforming memory access addresses within instructions executable by a processor – i.e., assembly code or byte code level instructions. There is a world of difference between the scope and content of Savitch and the environment relating to claims 2 and 10. Even if the skilled person were able to make the huge leap from C++ programming to memory address transformation for executable instructions, Savitch still teaches no more that using a stack to reverse the relative order of bytes (letters) within a particular word, whereas claims 2 and 10 require "where said any two bytes are spaced apart in memory by a plurality of words".

Other Cited Art

We have considered the other prior art documents cited by the Examiner on pages 14 and 15 of the Office Action. Tannenbaum provide no more than a basic description of the

differences between big endian and little endian conventions to confirm the understating already put forward by the Applicant and the Examiner. US 5,907,865 (Moyer), US 5,398,328 (Webber) and US 5,828,884 (Lee) each concern mechanisms to convert byte ordering with a word between big endian and little endian format which apply only on a word by word basis, like Loen.

New Claims

In light of our ongoing communications with the Examiner, Applicant introduces new claims 14 and 15 with a view to overcoming all of the objections that have been raised.

The method of independent claim 14 makes clear that actions which are performed in an initial translation (or compilation) phase, separate from the actions performed later in an execution phase. The recited features are based particularly on original claims 1-4 and looking at paragraphs [0015] to [0024] of the present application as published (US 2004/0204929A1). Like in Loen and the other cited documents, the order of bytes within each word is reversed from little endian to big endian format or vice versa. However, the important difference is that the order of words within memory is also reversed. Looking at the example in paragraph [0019] of the published application, a little-endian architecture would normally address memory from location "0" upwards to location "23". But, in the claimed invention, the big-endian architecture uses transformed memory access addresses within instructions such that the memory is now addressed downwards within this particular range from location "23" towards location "0". By transforming not only the location of bytes within words, but also the relative order of words, the correct data storage is preserved even though a wrong-endian processor is addressing the memory.

Claim 14 recites first a translation phase where the current program is allocated an area of memory to work in and, for this translation instance, a relatively simple calculation is performed that will transform any given subject address from a subject instruction into a transformed address in an output instruction. Such instructions are then available for execution using the transformed addresses, in the later execution phase.

Dependent claim 15 refines this method based particularly on paragraph [0043] of the published application wherein constant terms of the address transformation are folded in the

translation phase to provide partially resolved transformed memory access addresses in the output instructions. Then, in the execution phase, fully resolved transformed memory access addresses are calculated by calculating the remaining variable terms. As explained particularly at paragraph [0043], this folding feature minimizes the overhead at run time in the execution phase, even though a big-endian processor is working on little-endian data in the memory. The feature of claim 15 is yet another novel and inventive feature of the present invention over the available prior art.

In view of the above amendments and remarks, Applicant believes the pending application is in condition for allowance. Please apply any charges not covered, or any credits, to Deposit Account No. 08-0219.

In the event that there are any objections outstanding, we would welcome a telephone call from the Examiner to the undersigned. As indicated to the Examiner previously, the Applicant requests a personal interview with the Examiner so that prosecution of the application might be resolved quickly and efficiently.

Dated: June 19, 2006

Respectfully submitted,

By _____

Ronald R. Demsher
Registration No.: 42,478
WILMER CUTLER PICKERING HALE AND DORR LLP
60 State Street
Boston, Massachusetts 02109
(617) 526-6105
Attorney for Applicant